Тема
«Основы Java»

Состав
1. Отличия Java от других языков.
2. Компиляция и исполнение Java-кода.
3. Объектно-ориентированное программирование:
   3.1 Абстракция
   3.2 Наследование
   3.3 Инкапсуляция
   3.4 Полиморфизм

Отличия Java от других языков
1. Java является одновременно и компилируемым и интерпретируемым языком (концепция байт-кода, см. раздел «Компиляция и исполнения Java-кода»).
2. В Java реализована сборка мусора (будет рассмотрено на занятии «JVM, JIT, GC»).
3. Java свободно переопределяет порядок операций и очередность инструкций потоков по своему усмотрению (но в рамках контрактов модели памяти; рассматривается в занятии «Модель памяти Java»).
4. Java может свободно подменять некоторые инструкции в коде на более оптимальные (будет рассмотрено на занятии «JVM, JIT, GC»).
5. В Java реализована JIT-компиляция (будет рассмотрено на занятии «JVM, JIT, GC»).

Вывод- Java, это не просто еще один синтаксис для кода, это платформа.

Компиляция и исполнение Java-кода
Java-программа компилируется не в машинный язык, а в машинно-независимый код низкого уровня, байт-код. Далее байт-код выполняется виртуальной машиной.

Для выполнения байт-кода обычно используется интерпретация, хотя отдельные его части для ускорения работы программы могут быть транслированы в машинный код непосредственно во время выполнения программы по технологии компиляции «на лету» (Just-in-time compilation, JIT). Для Java байт-код исполняется виртуальной машиной Java (Java Virtual Machine, JVM).

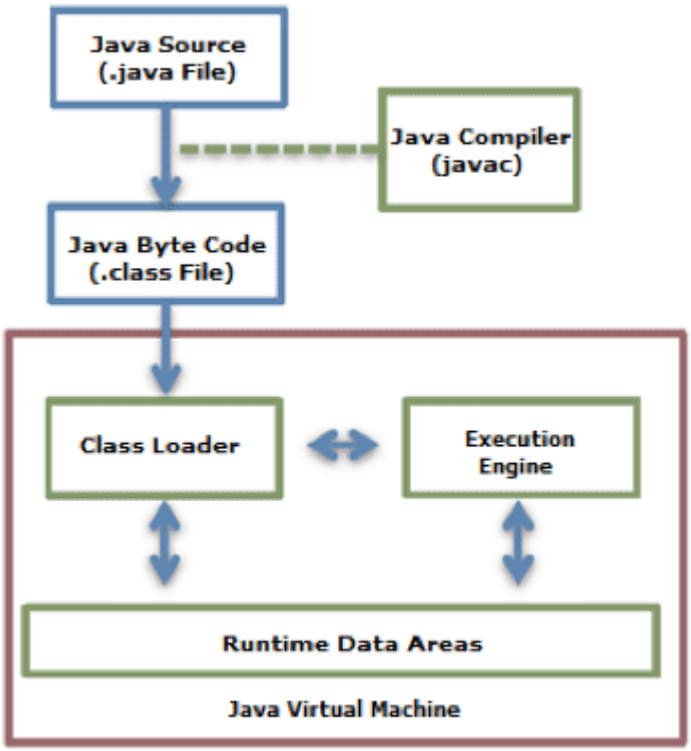
Подобный подход в некотором смысле позволяет использовать плюсы как интерпретаторов, так и компиляторов.

Недостатки:

- необходима программа - интерпретатор
- медленнее компилируемых программ
- большие требования к ресурсам
- требование корректности исходного кода (при внесении изменений требуется перекомпиляция кода)

Достоинства:

- компактность
- НЕзависимость от ОС (переносимость)
- быстродействие



ООП в Java

Объектно-ориентированное программирование — это шаблон проектирования ПО, позволяющий решать задачи разработчика с точки зрения взаимодействия объектов. Основными понятиями ООП являются класс и объект, при этом центральным из них — **объект**.

Объект — это сущность, экземпляр класса, которой можно посылать сообщения и которая может на них реагировать, используя свои данные.

Столпы ООП:

- Наследование
- Инкапсуляция
- Абстракция
- Полиморфизм

Абстракция

Абстрагирование — это способ выделить набор значимых характеристик объекта, исключая из рассмотрения не значимые Соответственно, **абстракция** — это набор всех таких характеристик.

Другими словами, абстракция это обобщение. Абстрактные классы в Java помечаются служебным словом **abstract.** Абстрактный класс, может содержать абстрактные методы, при этом все абстрактные методы должны быть переопределены в **не абстрактном** потомке абстрактного класса в обязательном порядке.

Также, к абстракции в Java относятся **интерфейсы.** Интерфейсы в Java помечаются служебным словом **interface.** Интерфейсы могут содержать абстрактные методы(методы без реализации), с Java 8 интерфейс может содержать реализацию метода по-умолчанию(default) и статические методы. Все методы в интерфейсе имеют модификатор доступа - **public**

Наследование

Наследование — это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником или производным классом.

Наследование в Java обозначается служебными словами: **extends**(если класс наследуется от другого класса) и **implements**(если класс реализует интерфейс).

Множественное наследование(multiple inheritance) в Java запрещено. **НО**, класс может реализовывать несколько интерфейсов. Количество интерфейсов, которые может реализовывать класс в Java, формально, не ограничено. Количество интерфейсов, может быть ограничено только размером заголовка класса в байт-коде.

Инкапсуляция

Инкапсуляция — это свойство системы, позволяющее объединить данные и методы, работающие с ними в классе, и скрыть детали реализации от пользователя.

Инкапсуляция позволяет управлять доступом к методам и полям объекта или класса, соответственно, то, что нужно, можно скрыть и оставить только то, что необходимо, для пользования объектом извне.

В Java инкапсуляция реализуется с помощью модификаторов доступа. В Java их 4 основных:

- **public** - этот модификатор доступа, предполагает доступ извне к методам и полям всем;
- **protected** - предполагает доступ к методам и полям самому классу, а также всем потомкам класса;
- **default(или отсутствие модификатора доступа)** - предполагает доступ к полям и методам всем, кто находится в одном пакете с искомым. Часто, этот модификатор доступа называют *package private*;
- **private** - скрывает от всех поля и методы объекта или класса. Поля и методы доступны только внутри самого объекта или класса.

Полиморфизм

Полиморфизм — это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

В программировании полиморфизм используется для того, чтобы сделать приложения более модульными и расширяемыми. Вместо беспорядочных условных предложений, описывающих различные направления действия, вы создаете взаимозаменяемые объекты, которые подбираете согласно своим нуждам. Это основная задача полиморфизма.

Полиморфизм позволяет работать с несколькими типами таким образом, как будто это один и тот же тип. И поведение объектов в данном случае будет разным и зависит от того, к какому типу они принадлежат. В общем, полиморфизм указывает, какую версию метода текущего объекта необходимо

запустить. Также полиморфизмом называют способность функции обрабатывать данные разных типов.

В Java полиморфизм реализуется благодаря позднему связыванию, когда тип конкретного “обобщенного” объекта определяется в рантайме, т.е. во время выполнения программы.